Lab 2 PSM, Trigger, Privileges

Goal:

After this lab, student should know how to

1. Create and use Stored procedure and function
2. How to create and use triggers
3. How to grant and revoke permissions

References

1. **MySQL reference**

https://dev.mysql.com/doc/refman/8.0/en/sql-statements.html

Software you need:

- MySQL command line or MySQL workbench.

Lab Report:

- You need to include the SQL statements and output for each question. You can copy and paste the statement and output or screenshot output. If the output of above is not sufficient to show that work has been done correctly, you also need to design some test cases to proof that your code(function, procedure, triggers, etc) does work correctly.

Lab Report template:

- For your convenience, a report template (Lab2_PSM_Report_template.doc) has been created for you. Please use it.

Part 1. PSM (30 points)

1. Setup:
   1.1. Create three tables:

```
lab2_course( id char(6) , name char(30), credit int)
lab2_student(id char(10), name char(10))
lab2_takes(id char(10), course_id char(6), grade char)
      id refs lab2_student(id)
            course_id refs lab2_course(id)
```

Please note: If you are putting all statements in a script, it would be helpful to include "drop table statement" before the "create table statement". This way you can rerun the script to recreate all tables. But if table does not exist yet, the drop table will fail. So use "drop table if exists" instead:

```
drop table if exists lab2_course ;
create table lab2_course (
      …
);
```

Include the SQL statements in the report

   1.2. Insert data for two students and two courses.

```
Courses:
      CS4421, "Database",  3 credits
      CS4461, "Network", 3 credits
Students:
      S001, Alice
      S002, Mike
```

Include the SQL insert statements in the report

2. Create a procedure named `enroll(student_id, course_id)` to enroll student into a class. If you want to edit an existing procedure, you need to drop it first using SQL command "`drop yourprocedurename`"

   2.1. Create procedure

   The default delimiter to separate SQL statements in MySQL is semicolon (;). But semicolor (;) is used in the create procedure statement, we need to set the delimiter to be something else. You may use // or $$ or any other strings that you don't expect to use in your procedure or functions. Since we are used to use ; as statement delimiter, let's change the delimeter back to ; after the procedure has been created.

   If you use MySQL Workbench, both semicolon (;) and ($$) can be used as delimiters. If you want to switch to //, please see https://dev.mysql.com/doc/workbench/en/wb-preferences-general-editors.html

```
delimiter //

drop procedure if exists enroll //
create procedure enroll (id char(10), course_id char(6) )
begin
insert into lab2_takes values (id, course_id, Null);
end //

delimiter ;
```

If you need to recreate the procedure, you need to drop it first. To drop a procedure, use "drop procedure" statement.

2.2. Call procedure

```
call enroll('S001','CS4421');
```

then use select statement to check whether the data in the database has been inserted correctly.

2.3. Grant read permission of the table lab_takes and execute permission to procedure enroll() to some student in the class. Then have them to check if they could read the data and execute your procedure.

You do:

```
grant select on lab2_takes to 'someone'@'%'
grant execute on procedure enroll to 'someone'@'%';
```

The student who was granted the permission do to confirm that they could execute the procedure successfully.

```
select * from yourdatabase.lab2_takes;
call yourdatabase.enroll('S002','CS4461');
select * from yourdatabase.lab2_takes;
```

3. Create a function `enrolled(course_id)` to return how many students are currently enrolled in the course

3.1. Create the function

```
delimiter //

drop function if exists enrolled //

create function enrolled(c_id char(6))
    returns int
begin
   declare total int;
    select count(*) into total from lab2_takes where course_id= c_id;
    return total;
end //
delimiter ;
```

3.2. Call function enrolled and check if it returns the correct result

```
select enrolled('cs4421');
select enrolled('cs4461');
```

Part 2. Trigger (20 points)

1. Create a trigger to update the total credit when grade is updated

1.1. Add column total_credits.

```
alter table lab2_student add total_credits int default 0;
```

1.2. Create trigger

Please pay attention to how predefined tuple variable OLD and NEW are used.

```
delimiter //
drop trigger if exists update_credits;
create trigger update_credits
   after update on lab2_takes
   for each row
begin
   if OLD.id = NEW.id and OLD.course_id = NEW.course_id
      and ( OLD.grade is null or OLD.grade = 'F')
      and NEW.grade is not null
      and NEW.grade != 'F' then
              update lab2_student
              set total_credits = total_credits +
                 (select credit
                  from lab2_course
                  where id = OLD.course_id )
              where id=NEW.id;
   end if;
end//
delimiter ;
```

1.3. Choose a student and a course that the student is taking. Then update the grade (from NULL or F to A,B,C,D grade) and check if `total_credits` is updated accordingly. Something like this:

```
select * from lab2_student where id=…;   //pay attention to the total_credits

select * from lab2_takes where id=…;

update lab2_takes set grade =…. where id=… and course_id=…

select * from lab2_takes where id=… ;

select * from lab2_student where id=…; //the total_credits should have  been
changed by the trigger.
```